NON-DISRUPTIVE RECONFIGURATION OF A PUBLISH/SUBSCRIBE SYSTEM

Cross-Reference to Related Applications

[0001] This application is a continuation of prior Application No. 09/280,534, filed March 30, 1999, which is hereby incorporated herein by reference in its entirety.

[0002] This application contains subject matter which is related to the subject matter of the following United States patent applications, which are assigned to the same assignee of this application. Each of the below-listed applications is hereby incorporated herein by reference in its entirety:

[0003] "Routing Messages Within A Network Using The Data Content Of The Message," by Chandra et al., U.S. Patent No. 6,091,724, issued July 18, 2000;

[0004] "Message Logging For Reliable Multicasting Across A Routing Network," by Banavar et al., serial no. 09/281,421, filed March 30, 1999, (attorney docket no. YO998-525);

[0005] "Message Sequencing For Ordered Multicasting Of A Message Across A Routing Network," by Banavar et al., serial no. 09/280,530, filed March 30, 1999, (attorney docket no. YO998-526); and

[0006] "Quiescent Reconfiguration Of A Routing Network", by Miller et al., serial no. 09/282,101, filed March 30, 1999, (attorney docket no. YO999-527).

Technical Field

[0007] This invention relates, in general, to reconfiguring a routing network and, in particular, to non-disruptively reconfiguring a publish/ subscribe system without losing or reordering messages of the system during the reconfiguration.

Background of the Invention

[0008] Many network environments enable messages to be forwarded from one site within the network to one or more other sites using a multicast protocol. Typical multicast protocols send messages from one site to one or more other sites based on information stored within a message header. That is, each message has two components: the message header, which includes the routing information, including destination addresses or a predefined group name that is associated with a fixed list of destinations, and a data content, which is the data of the message. The routing information is read from the message header and is used to send the data content of the message to the specified destinations.

[0009] One example of a system that conventionally includes such a network environment is a publish/subscribe system. In publish/subscribe systems, publishers post messages and subscribers independently specify categories of messages in which they are interested. The system takes the posted messages and includes in each message header the destination information of those subscribers indicating interest in the particular message. The system then uses the destination information in the message to forward the message through the network to the appropriate subscribers.

[0010] More particularly, a publish/subscribe system includes a network of message routers (or simply routers) connected via links in an arbitrary graph topology. A number of clients connect to the periphery of this router network and either publish or subscribe to messages. A message includes a number of attributes, which are name-value pairs.

[0011] The problem addressed by the present application is how to handle changes of topology in such a routing network. The need for changing a topology arises from a number of conditions, including: (1) the need for system maintainers to move routers on-line and off-line; (2) the growth of the network; and (3) changes in speed that alter the optimum spanning trees of the network; etc. An assumption is made that a configuration manager has made a decision to effect a particular topology change (i.e., a reconfiguration). The present invention is directed to providing a technique by which the nodes of the router network execute a reconfiguration decision, eventually resulting in a new state of the network in which messages are forwarded using a new spanning tree.

Summary of the Invention

[0012] The shortcomings of the prior art are overcome and additional advantages are provided through the provision of a method of reconfiguring publish/subscribe systems. The method includes, for instance, initiating a reconfiguration of a publish/subscribe system; and reconfiguring the publish/subscribe system, wherein one or more messages of the publish/subscribe system are not lost during the reconfiguring.

[0013] In one embodiment, the reconfiguring is non-disruptive to the publish/subscribe system.

[0014] As one example, the reconfiguring includes changing from a first routing path between one node of the publish/subscribe system and another node of the system to a second routing path between the one node and the another node.

[0015] In one embodiment, the publish/subscribe system includes an ordering requirement for delivery of one or more messages from at least one node to at least one other node of the publish/subscribe system, and the reconfiguring preserves the ordering of delivery of the one or more messages.

[0016] In a further example, the method includes forwarding a message from at least one node of the publish/subscribe system to at least one other node of the system, after the reconfiguration is initiated. Additionally, another message is forwarded from at least one node of the system to at least one other node, wherein the another message is forwarded using a different routing path than the message.

[0017] In another aspect of the present invention, a system of reconfiguring publish/subscribe systems is provided. The system includes, for instance, means for initiating a reconfiguration of a publish/subscribe system; and means for reconfiguring the publish/subscribe system, wherein one or more messages of the publish/subscribe system are not lost during the reconfiguring.

[0018] In yet a further aspect of the present invention, an article of manufacture including at least one computer usable medium having computer readable program code means embodied therein for causing the reconfiguring of publish/subscribe systems is provided. The computer readable program code means in the article of manufacture includes, for instance, computer readable program code means for causing a computer to initiate a reconfiguration of a publish/subscribe system; and computer readable program code means for causing a computer to reconfigure the publish/subscribe system, wherein one or more messages of the publish/subscribe system are not lost during the reconfiguring.

[0019] The present invention advantageously provides for dynamic reconfiguration of a system. That is, the reconfiguration is accomplished without shutting down the network. In particular, it is not necessary to quiesce (i.e., block senders of messages from introducing new messages), while the reconfiguration is taking place. The execution of a reconfiguration is not directly visible to either publishers or subscribers. The publishers and subscribers continue to publish and receive messages as if no reconfiguration is taking place or has taken place.

[0020] The non-disruptive reconfiguration capabilities of the present invention have particular application in large, continuously available broker networks in which change and evolution are inevitable and service disruption is intolerable.

[0021] Additional features and advantages are realized through the techniques of the present invention. Other embodiments and aspects of the invention are described in detail herein and are considered a part of the claimed invention.

Brief Description of the Drawings

[0022] The subject matter which is regarded as the invention is particularly pointed out and distinctly claimed in the claims at the conclusion of the specification. The foregoing and other objects, features, and advantages of the invention are apparent from the following detailed description taken in conjunction with the accompanying drawings in which:

[0023] FIG. 1 depicts one example of a distributed network incorporating and using the non-disruptive reconfiguration capabilities of the present invention;

[0024] FIG. 2 depicts one example of a spanning tree used in accordance with the principles of the present invention;

[0025] FIG. 3 depicts one embodiment of a distributed router network, which is to undergo non-disruptive reconfiguration in accordance with the present invention;

[0026] FIG. 4 depicts one embodiment of a router for use, in accordance with the principles of the present invention, in a router network such as the one depicted in FIG. 3;

[0027] FIGs. 5a-5g depict examples of data structures associated with a router of a network to undergo non-disruptive reconfiguration, in accordance with the principles of the present invention;

[0028] FIGs. 6a-6c depict examples of data structures associated with a client of a network to undergo non-disruptive reconfiguration, in accordance with the principles of the present invention;

[0029] FIGs. 7a-7d depict examples of data structures associated with a special node of a network to undergo non-disruptive reconfiguration, in accordance with the principles of the present invention;

[0030] FIGs. 8a-8e depict examples of headers of various messages used in accordance with the principles of the present invention;

[0031] FIG. 9 depicts one embodiment of the logic associated with processing a reconfiguration request, in accordance with the principles of the present invention;

[0032] FIGs. 10a-10b depict one embodiment of the logic associated with processing a CS-message, in accordance with the principles of the present invention;

[0033] FIG. 11 depicts one embodiment of the logic associated with processing an SC-message, in accordance with the principles of the present invention; and

[0034] FIG. 12 depicts one embodiment of the logic associated with processing a CC-message, in accordance with the principles of the present invention.

Best Mode for Carrying Out the Invention

[0035] In accordance with the principles of the present invention, a reconfiguration capability is provided for distributed network environments, such as those included in publish/subscribe systems. As one example, a publish/subscribe system is non-disruptively reconfigured such that no messages are lost during the reconfiguration. Further, the reconfiguration is performed without quiescing the system (i.e., without having to suspend routing of messages). Additionally, properties of various messages within the system are preserved. For example, various messages have an ordering requirement associated therewith, and that ordering requirement is preserved during the reconfiguration. The present invention advantageously enables reconfiguration to be performed without affecting the qualities of service guaranteed to the clients.

[0036] One example of a distributed network 100 incorporating and using the reconfiguration capabilities of the present invention is depicted in FIG. 1 and described in detail below. Network 100 includes, for instance, a plurality of computing units 102 coupled to one another via links 104.

[0037] Each link couples the computing units in the network, and each computing unit may have any number of links connected to it. Each link is bidirectional (i.e., a computing unit may send and receive messages on the link.) Each computing unit in the network is either a client computer (represented by the smaller ovals, such as those having addresses 101a, 101b), meaning that it has originated messages or requested to receive messages; or it is a router computer (represented by the larger ovals, such as 108a, 108b), meaning that it forwards messages received on one network link onto other links on the way to the client computer(s). The clients are collectively referred to herein as clients 101 and the routers are collectively referred to as routers 108. (For purposes of this discussion, if a single computing unit serves both as a router and as a client, these two separate functions will be considered as two computing units connected by a link.)

[0038] Each computing unit can be any type of computing unit that can be included in a network. For example, it can be an RS/6000 computing node or any other of various types of computing nodes, processors, computers or systems. The network can also include different types of computing units coupled to one another via the links. The links include, for instance, TCP connections over IP links, as only one example.

[0039] Distributed network 100 can be included in various systems that require the passing of messages or data. These systems include, for instance, publish/subscribe systems. As used herein, a publish/subscribe system includes any system that multicasts messages to one or more nodes of the system.

[0040] Examples of publish/subscribe systems include group-based systems in which each message is initially assigned to one of a small number of groups representing a set of destinations; and content-based systems in which messages are routed from publishers to subscribers based on the content of the data within the message. With data-content routing, a message does not need to include destination information, such as destination addresses or a group destination name. Instead, data within the message is used to traverse a data structure to determine the link or links over which the message is to be forwarded in order to reach the consumers (subscribers or clients) interested in the message.

[0041] Content-based publish/subscribe systems improve the degree of decoupling between publishers and subscribers. In content-based publish/subscribe systems, subscriptions are specified in terms of predicates on the posted data, rather than in terms of subject identifiers supplied by the publisher. One example of a content-based publish/subscribe system is described in co-pending U.S. Patent No. 6,216,132, issued, April 10, 2001, entitled "Method And System For Matching Consumers To Events," Astley et al., which is hereby incorporated herein by reference in its entirety. Some of the examples described herein are with reference to a content-based subscription system. However, these are only examples. The present invention can be employed with other types of systems without departing from the spirit of the present invention.

[0042] As noted above, the present invention is applicable to group-based (i.e., subject-based) publish/subscribe systems. As a further example, the present invention is applicable to any systems in which the topology is relatively fixed. That is, at any time, each node in the network knows a single best path for a particular kind of message originating at node N1 and destined for node N2. The rules for computing this best path changes infrequently. Further, requirements for ordered delivery can be implemented by designating a small number of nodes to support strictly first in-first out (FIFO) message streams with the other nodes. In particular, it is assumed that message delivery is required to be FIFO between each client and each of a set of k nodes called special nodes s1,...sk, but that other message delivery is not required to be ordered.

[0043] Systems implementing reliable routing protocols and ordered delivery protocols are respectively described in "Message Logging For Reliable Multicasting Across A Routing Network," by Banavar et al., co-filed herewith, serial no. 09/281,421, filed March 30, 1999, and "Message Sequencing For Ordered Multicasting Of A Message Across A Routing Network," by Banavar et al., co-filed herewith, serial no. 09/280,530, filed March 30, 1999; where the "logger" nodes of the reliable routing protocol and the "sequencer" node of the ordered delivery protocol play the role of the special nodes herein.

[0044] In one embodiment of the invention, each router 108 of network 100 (FIG. 1) has associated therewith a spanning tree, which lays out the best path (according to some criterion, such as latency) from the router to one or more clients 101. In this embodiment, it is assumed that routers agree on a common criterion for measuring distance between nodes in the network. There may in fact be multiple spanning trees. For example, alternative spanning trees may specify either backup routes, or peak load routes. Herein, it is assumed that one spanning tree is in effect for the routing of any particular message.

[0045] One example of a spanning tree, which is associated with a router, is depicted in FIG. 2. As shown in FIG. 2, there is a path from router 102a to every other node in the spanning tree. A message to be routed from router 102a to one or more of the other nodes is routed via one or more of the links associated with router 102a, i.e., links 1-3. For example, if a message is to be routed from node 102a to node 101c, then link 2 is used. As a further example, if a message is to be forwarded toward client 101a, then link 3 is used.

[0046] One embodiment for building a spanning tree from a network topology, that is an arbitrary graph, is described in detail in <u>Introduction to Algorithms</u>, by Cormen, Leiserson, Rivert, Chapter 24, pp 498-513, Published by MIT Press (1990), which is hereby incorporated herein by reference in its entirety.

[0047] By way of example, FIG. 3 depicts one embodiment of a publish/subscribe system, generally denoted 300, to employ non-disruptive reconfiguration in accordance with the principles of the present invention. System 300 includes a network of routers 302 connected via links 303 in an arbitrary graph topology. A number of clients connected to the periphery of this router network either publish messages or subscribe to messages. Those clients publishing messages comprise publishers 304, while those clients subscribing to messages comprise subscribers 306. The router network is responsible for routing messages from a publisher 304 to interested subscribers 306 based, for example, on matching messages to subscription predicates.

[0048] This protocol, known as content-based routing, is described in detail in the initially-incorporated co-pending patent application entitled “Routing Messages Within A Network Using The Data Content Of The Message.” As noted, from each router node at which a publisher is present, the system computes and stores a spanning tree to reach every other node in the network. All published messages from the publisher follow the paths in that spanning tree, with each router node performing enough matching to determine which of its child routers should receive the message.

[0049] Within a publish/subscribe system, the present invention can be employed, in one example, with a network achieving reliability of routed messages by saving messages to persistent storage within the network (and then retrieving and redelivering the message whenever there is a failure in the network). Such a reliable routing network is described in the above-incorporated, co-filed patent application entitled “Message Logging For Reliable Multicasting Across A Routing Network.” In accordance with that invention, publishers and subscribers that need reliability of messages may specify a quality of service parameter, e.g., “uniform delivery”. Uniform delivery is provided for ensuring delivery of a message to all active subscribers notwithstanding failure in the network, e.g., the routers, or the links. One or more special nodes 310 in router network 300 are designated logging node(s) or logger(s) and support the ability to log messages to stable storage. When there is at least one subscriber needing logging, the routing algorithm ensures that messages are routed to the logger.

[0050] Alternatively, or in further combination, the present invention can be employed in another example with a network achieving ordering of routed messages by sequencing messages within the router network itself through assignment of a sequence number at a designated sequencing node of the network. Publishers and subscribers that need ordering of messages may specify a quality of service parameter called "totally ordered". Total ordering is provided by ensuring ordered delivery of a message to all active subscribers. Such message sequencing within the router network is described in the above-incorporated, co-filed application entitled "Message Sequencing For Ordered Multicasting Of A Message Across A Routing Network." When employing sequencing, a special node 310 in router network 300 is designated a sequencing node or sequencer. Node 310 supports the ability to sequence messages received into the routing network. When there is at least one subscriber needing total ordering, the routing algorithm guarantees that messages are routed to the sequencer.

[0051] In one example, each special node is coupled to a configuration manager 312. (In another embodiment, one or more, but not necessarily all, special nodes are connected to the configuration manager.) The configuration manager is, for instance, responsible for making the decision to effect a particular topology change, i.e., a reconfiguration. A configuration manager is, for instance, a "network system service" that is responsible for maintaining information about the structure, function and status of a network system. This includes information such as the network topology, nodes with certain properties (such as the special nodes), quality of service offered to the nodes in the network, etc. A system's configuration may be statically fixed or dynamically changing. A configuration manager may be centralized or distributed. Network system management products such as IBM's Tivoli TME-10 contain configuration management components.

[0052] To summarize, in one example, non-disruptive reconfiguration pursuant to the present invention can be employed in a router network of a publish/ subscribe system which utilizes at least one special node to facilitate either, or both, logging of messages or sequencing of messages with the network itself.

[0053] In the above example, client nodes 304, 306, router nodes 302 and special nodes 310 are shown as separate physical computers connected via bidirectional links. However, in another example, all three types of nodes (or a subset thereof) can be located within a single physical computer and connected by one or more virtual links.

[0054] Further details of one embodiment of a content-based router node, used in accordance with the principles of the present invention, are described with reference to FIG. 4. When a message arrives at router 400, it is stored into a message table 402 using a unique message identifier. Information stored includes a source node identification, as well as the neighboring nodes to which the message is to be forwarded. These neighboring nodes are calculated by a content routing computation component 404 after the message has been received. Computation component 404 takes the message and based upon stored subscriptions returns a set of destinations or links upon which the message should be forwarded. Again, in one embodiment, this computation is content dependent and can be accomplished as described in the above-incorporated co-pending application. However, this component can also be implemented in other ways, e.g., subject-based routing.

[0055] In one embodiment, to facilitate reliable routing of messages, router 400 also includes a reliable routing component 406, a logging acknowledgment (LACK) received table 408 and a LACK send table 409, which are described in the above-incorporated, co-filed application entitled "Message Logging For Reliable Multicasting Across A Routing Network."

[0056] If ordered routing of messages is employed, then router 400 would also include recovery data, including a latest sequence number received and linked node tables (not shown) such as described in the above-incorporated, co-filed application entitled "Message Sequencing For Ordered Multicasting Of A Message Across A Routing Network." Again, the non-disruptive reconfiguration approach of this invention can be employed in a routing network of a publish/subscribe system employing data content messaging, reliable routing of messages, or sequencing of messages, either individually or in combination as will be apparent from the following description.

[0057] Various nodes within the network have data structures associated therewith that are used during the reconfiguration of the present invention. These data structures are described below with reference to FIGs. 5a-7d.

[0058] For example, with reference to FIGs. 5a-5b, each router node has two routing tables associated therewith, a Table-0 (500a of FIG. 5a) and a Table-1 (500b of FIG. 5b). (Although in the embodiment described herein, each router has data structures associated therewith, in another embodiment, one or more of the routers may not have such data structures. The same is true for the other nodes described below.) In one example, the routing tables are located within storage of the corresponding router node. Table 500a includes, for instance, one or more destination nodes 502a, and at least one path 504a for each destination node. For instance, Table 500a includes an entry for each client computer in the network. Each entry of the routing table associates a client address with the identifier of the network link constituting the next segment on the path in the spanning tree from the router to the client. For a router with d network links, each such link identifier is an integer between 1 and d. For instance, the client having address 101a has a corresponding link identifier of 3 (see FIG. 2) in the router table for router 102a.

[0059] Similarly, Table-1 includes one or more destination nodes 502b and at least one path 504b for each destination node. Initially, one of the tables, e.g., Table-0, is configured by its corresponding router with information provided by the configuration manager. That is, the routing table is constructed via information from the network topology (e.g., the client addresses) and hence from the corresponding spanning tree (e.g., the link identifiers), in a known manner. The other table, e.g., Table-1, is initially empty.

[0060] At a topology change, i.e., a reconfiguration, configuration manager 312 distributes information allowing each router 108 to configure a new topology in Table-1. For a period of time, some messages are routed using Table-0 (the old table), while others use Table-1 (the new table). Eventually, the configuration stabilizes to a point where all routers are using the new table for all messages. After this time, it is possible for the configuration manager to initiate a new reconfiguration. When the next reconfiguration occurs, Table-0 is the new table and Table-1 is the old table. Thus, on odd-numbered reconfigurations, Table-1 is the new table, and on even-numbered reconfigurations, Table-0 is the new table.

[0061] The current table is indicated by a current table indicator 506 (FIG. 5c) stored within, for instance, the routing node. In one example, current table indicator 506 is a binary indicator. Initially, it is set to 0, and is flip-flopped each time a reconfiguration takes place.

[0062] In addition to the above routing tables, each router has a number of other data structures associated therewith. For example, each router node keeps an outbound table vector 508 (FIG. 5d) of k bits, where k is equal to the number of special nodes. Bit i in this vector is 1, if Table-1 is to be used on messages sent from special node si, and it is 0, if Table-0 is to be used. Each router node also has an inbound table vector 510 (FIG. 5e) of k bits. Bit i in this vector is 1, if Table-1 is to be used on messages sent through this router towards special node si, and it is 0, if Table-0 is to be used.

[0063] Additionally, each router node has a switch acknowledgment count vector 512 (FIG. 5f) of k counts. A nonzero count c in position i implies that a switch message (described below) has been received on a spanning tree originating at special node si and the router has forwarded the switch message to the nodes below it, but has not yet received a switch-ack message from c of these nodes. Finally, each router node with attached clients keeps an Endstream-sent vector 514 (FIG. 5g) of k bits. The i-th bit is on, if an Endstream-message (described below) has been sent to the i-th special node on behalf of one of the clients attached to this router. Initially, all bits are set.

[0064] In addition to the data structures for router nodes, each client node also has one or more data structures associated therewith. For example, each client node has a held message queue 600 (FIG. 6a) for each special node of the system. Held message queue 600 holds SC-messages (described below) received at the client from a special node along a path determined by a new topology that are being held until all earlier SC-messages sent from that special node along the path determined by the old topology have been received. Each client node also includes a delivery queue 602 (FIG. 6b) of SC-messages ready to be received by clients. Each client node keeps a table vector 604 (FIG. 6c), which indicates for each special node which table represents the new path and which represents the old path. Initially, this is set to 0 for each special node.

[0065] Each special node also has one or more data structures associated therewith. For example, each special node has a held message queue 700 (FIG. 7a) for each client node, which has delivered a CS-message (described below) to that special node along a new topology prior to the special node receiving all earlier CS-messages from that client along the path determined by the old topology. It additionally keeps a delivery queue 702 (FIG. 7b) of CS-messages ready to be delivered to the special node. Each special node keeps an indicator 704 (FIG. 7c) to determine whether the latest table is Table-0 or Table-1. Further, each special node keeps a list 706 (FIG. 7d) of clients that have completed sending FIFO streams from the previous table; initially that is set to the value ALL.

[0066] The above-described data structures are used, in accordance with the principles of the present invention, during the routing of messages from one or more nodes of a publish/subscribe system to one or more other nodes. Examples of messages employed with the present invention include reconfiguration messages sent from the configuration manager to one or more special nodes of the system, when a reconfiguration is to take place; a CS-message delivered FIFO between a client and a special node; an SC-message delivered FIFO between a special node and a client; a CC-message delivered multicast, unordered between two clients; an Endstream-message sent by a client to a special node (and delivered FIFO), when it is the end of a sequence of messages sent to the special node on the old path; and a switch message sent to indicate the switching from an old path to a new path.

[0067] Each of the messages described above, except for the reconfiguration request, includes a message header that provides information used during routing. Examples of some of the data fields associated with the various message headers are described below with reference to FIGs. 8a-8e.

[0068] Referring to FIG. 8a, as one example, a switch message header 800 or a switch-ack (switch acknowledgment) message header 800 includes, for instance, a special node origin field 802 indicating the index (i.e., the address) of the special node from which the switch message originated; and a table number field 804 indicating which routing table (Table-0 or Table-1) is the current table.

[0069] A CS-message header 806 (FIG. 8b) includes, for example, a special node destination field 808 indicating the index of the special node to receive the message; a table number 810 indicating the current routing table; and an origin client field 812 specifying an address of the client originating the CS-message.

[0070] Referring to FIG. 8c, Endstream-message header 816 includes, for instance, a special node destination field 818 indicating the index of the special node to receive the message; a table number field 820 specifying the current routing table; and an origin client field 822 indicating an index of the client originating the message.

[0071] An SC-message header 830 (FIG. 8d) includes, for example, a destination client field 832 specifying the address of the client to receive the message; a table number field 834 indicating the current routing table; and a special node origin field 836 specifying an index of the special node originating the message.

[0072] Lastly, referring to FIG. 8e, a CC-message header 840 includes a table number field 842 specifying the current routing table.

[0073] Details of how the fields of the message headers are used during processing of the messages are described in detail below with reference to FIGs. 9-12.

[0074] Referring to FIG. 9, one embodiment of the logic associated with processing a reconfiguration request is described in detail. When a decision is made to reconfigure the system (e.g., change the network or routing topology), the configuration manager sends a reconfiguration request message to one or more (preferably each) of the special nodes of the system. The processing described below is with reference to one special node. However, similar processing is performed by each special node that receives the request.

[0075] Initially, the configuration manager sends a reconfiguration request to a special node, STEP 900. This is accomplished via, for instance, a multicast operation or a point to point operation. When the special node receives the message, it flips the polarity of its current table indicator 704, STEP 902. For example, if the indicator was set to 0, then it is changed to 1 indicating that routing Table-1 is the new current table.

[0076] Thereafter, the special node that received the reconfiguration request passes a switch message down the spanning tree towards clients 101, STEP 904. The switch message includes the new table indicator in table number field 804, and the index of the special node sending the switch message in special node origin field 802. The switch message also includes the contents of the designated table. The switch message is propagated through routers 108, and during propagation, the router node data structures are updated, STEP 906. For instance, at each router 108 that receives the message, the new table indicator is saved as indicator 506 (FIG. 5c). Further, outbound table vector 508 (FIG. 5d) of the router is updated by having the bit in the position corresponding to the special node index i (as specified in field 802) set to the value of table indicator 506. For example, if i is equal to 2, then the bit in position 2 of the vector is updated.

[0077] Propagation of the switch message through the routers includes, for each outbound link (e.g., all links except the one back towards the special node), propagating a copy of the switch message downward towards the clients, and updating switch acknowledgment count vector 512 (FIG. 5f). In particular, the position within the vector corresponding to the special node index i is set equal to the number of copies propagated.

[0078] Further, if the router node has clients, then an entry in bit position i of endstream-sent vector 514 (FIG. 5g) is cleared. When the switch message reaches a client, then the client node data structures are updated, STEP 908. For example, bit position i of table vector 604 (FIG. 6c) is set equal to the new table indicator, the messages on held queue 600 are transferred to delivery queue 602, and the held queue is emptied.

[0079] A switch-ack message is then propagated from the client back towards the special node, STEP 910. For example, when a router receives a switch-ack message for a special node i, it decrements the counter for position i in switch-ack count vector 512 (FIG. 5f). When the counter reaches 0, indicating that all of the intended clients have received the switch message from the special node and they have acknowledged the same, a switch-ack message is propagated up to the next hop on the path to the special node.

[0080] When a switch-ack message reaches a special node, it is propagated to the configuration manager, STEP 916. Additionally, the special node transfers any messages in held message queue 700 to the corresponding delivery queue 702, and resets list 706 to all, indicating that only the new path will be used for any further messages, until the next reconfiguration, STEP 918. When the configuration manager receives a switch-ack message from all of the special nodes, it then performs a sweep to check for straggler messages that are still using the old routing tables, STEP 920. There are a number of existing sweep techniques in the published literature which can be used for such purpose; one such technique is the Jefferson GVT technique, described in DR Jefferson "Virtual Time" ACM Transactions on Programming Languages and Systems, Vol 7 #3, July 1985, pp 404-425, which is hereby incorporated herein by reference in its entirety.

[0081] If stragglers exist, INQUIRY 922, then the configuration manager waits a predefined amount of time, STEP 924, and then performs a sweep, again, STEP 920. When it is confirmed that no stragglers exist, INQUIRY 922, processing of the reconfiguration request is complete, STEP 926.

[0082] After the configuration manager initiates a reconfiguration request and before the nodes update their data structures, the system is in a mode where there are two tables in effect, i.e., the old table and the new table. FIGs. 10-12 describe how processing is to be performed for specific messages, when the system is in such a mode.

[0083] As one example, FIGs. 10a-10b describe one embodiment of the processing associated with a CS-message, in accordance with the principles of the present invention. The logic is described with reference to one router receiving the message. However, each router that receives the message performs similarly.

[0084] Referring to FIG. 10a, initially, a router of the system receives a CS-message that is to be routed towards a special node, STEP 1000. When the router receives the message, a check is made to determine if the arriving CS-message is from a client, INQUIRY 1002. If it is from a client, then inbound table vector 510 (FIG. 5e) is examined in the bit position corresponding to the special node designated by special node destination field 808 supplied in the CS-message. As a result, either Table-0 or Table-1 is selected for routing the message towards the special node, STEP 1004.

[0085] Thereafter, a determination is made as to whether the corresponding bit in endstream-sent vector 514 (FIG. 5g) is set, INQUIRY 1006. If it is not set, then table indicator 820 in an Endstream message is set to the old table, STEP 1008, and the Endstream message is sent to the special node using the route defined by the old table (the opposite of the table selected by vector 510), STEP 1010. The manner in which the Endstream message is sent is similar to the manner in which the CS-message is sent. Additionally, the corresponding bit in Endstream-sent vector 514 is then turned on, STEP 1012. Thereafter, table indicator 810 in the CS-message is set to record which table was used, STEP 1014, and the CS-message is then forwarded to the next node, which is either another router 108 or the destination special node, STEP 1016.

[0086] Returning to INQUIRY 1006, if the endstream-sent vector bit is set, then processing proceeds to STEP 1014, in which the table indicator is set. Subsequently, the CS-message is forwarded to the next node, STEP 1016.

[0087] Returning to INQUIRY 1002, if the arriving message is not from the client, but from another router, STEP 1018, then table indicator 810 in the message is used to determine the route to take towards the special node, STEP 1020. Thereafter, the CS-message is forwarded to the next node, STEP 1016, which is either another router or the destination special node.

[0088] Subsequently, a determination is made as to whether the message is at the special node, INQUIRY 1022 (FIG. 10b). If not, then processing continues with STEP 1000. That is, the message is to be processed by the next router.

[0089] However, if the message has arrived at the special node, then table indicator 810 is compared against the latest table indicator 704, INQUIRY 1024. If the CS-message is for the old table, it is immediately queued to delivery queue 702, STEP 1026. Otherwise, a determination is made as to whether the origin client has finished its FIFO stream for the old table, as determined by list 706, INQUIRY 1028. If the client is on list 706, the message is queued to delivery queue 702, STEP 1026. Otherwise, it is queued to held message queue 700, STEP 1030. Processing of the CS-message is then complete.

[0090] Messages are transferred from held message queue 700 for the specified client to delivery queue 702, when an Endstream-message arrives at the special node or when the switch-ack is received. At that point, the held message queue is emptied, and the client is added to list 706. The messages on delivery queue 702 are delivered to the special node in the order that they are received on the queue.

[0091] One embodiment of processing SC-messages, in accordance with the principles of the present invention, is described with reference to FIG. 11. Again, the logic is described with reference to one router receiving the message. However, each router that receives the message performs similar logic.

[0092] Referring to FIG. 11, initially, a router receives a message, STEP 1100. When the router receives the message, a determination is made as to whether the arriving SC-message is from a special node, INQUIRY 1102. If so, then outbound table vector 508 is examined in the bit position corresponding to the special node designated in special node origin field 836. As a result, either Table-0 or Table-1 is selected for routing the message towards the client, STEP 1104. Thereafter, table indicator 834 is set to indicate the table to be used in subsequent hops toward the client, STEP 1106.

[0093] Returning to INQUIRY 1102, if the arriving SC-message is not from a special node, then it is assumed to be from another router, STEP 1108. Thus, table indicator field 834 in the message is used to determine which table is to be used to route the message toward the client, STEP 1110.

[0094] Subsequent to determining which table to use, STEP 1110, or subsequent to setting the table indicator, STEP 1106, the message is forwarded to the next node, which is either another router 108 or the destination client 101, STEP 1112. If the next node is not the client, INQUIRY 1114, then processing continues with STEP 1100. Otherwise, table indicator 834 in the message is compared with table vector 604 in the bit position corresponding to the special node origin specified in message field 836, INQUIRY 1116. If the message was sent over the old path (tables match), it is forwarded directly to delivery queue 602, STEP 1118. If it was sent over the new path (tables do not match), it is forwarded to held message queue 600, STEP 1120. Processing of the SC-message is then complete.

[0095] Messages on held message queue 600 are transferred to delivery queue 602, when a switch message is received at the client, as described above.

[0096] FIG. 12 depicts one embodiment of the processing associated with CC-messages, in accordance with the principles of the present invention. The processing is once again described with reference to one router, but each router receiving the message performs similarly.

[0097] A CC-message is multicast using a single path, either the old path or the new path, STEP 1200. When the message reaches the nearest router to the client delivering the CC message, that router uses table indicator 506 to decide which table to use, STEP 1202. That table indicator is then stored in table indicator field 842 of the CC-message, STEP 1204. The message is forwarded or multicast to the next hop or hops using that table. This continues until all destination clients have received the message.

[0098] Optimizations of the above techniques are possible. For instance, if the same message is multicast to a client or clients as a CC-message, and sent to a special node as a CS-message, it is possible for these two logical messages to be bundled into a single physical message. Other similar optimizations in which the information content of the defined messages are bundled together may be performed.

[0099] Described in detail above is the processing that takes place when a reconfiguration request is initiated. In accordance with the principles of the present invention, each of the different types of messages determines which path to use during the routing of the message. Some messages may be routed using the old path and some may be routed using the new path, until the system is stabilized. This allows reconfiguration to take place that is non-disruptive and does not require quiescing. Further, messages are not lost during the reconfiguration, and ordering of the delivery of the messages is preserved (reordering is not necessary, either).

[00100] The present invention can be included in an article of manufacture (e.g., one or more computer program products) having, for instance, computer usable media. The media has embodied therein, for instance, computer readable program code means for providing and facilitating the capabilities of the present invention. The article of manufacture can be included as a part of a computer system or sold separately.

[00101] Additionally, at least one program storage device readable by a machine, tangibly embodying at least one program of instructions executable by the machine to perform the capabilities of the present invention can be provided.

[00102] The flow diagrams depicted herein are just exemplary. There may be many variations to these diagrams or the steps (or operations) described therein without departing from the spirit of the invention. For instance, the steps may be performed in a differing order, or steps may be added, deleted or modified. All of these variations are considered a part of the claimed invention.

[00103] Although preferred embodiments have been depicted and described in detail herein, it will be apparent to those skilled in the relevant art that various modifications, additions, substitutions and the like can be made without departing from the spirit of the invention and these are therefore considered to be within the scope of the invention as defined in the following claims.